Structure Discovery in Sequentially Connected Data

Jeffrey A. Coble, Diane J. Cook, and Lawrence B. Holder

Department of Computer Science and Engineering
The University of Texas at Arlington
Box 19015, Arlington, TX 76019
{coble,cook,holder}@cse.uta.edu

Abstract

Much of current data mining research is focused on discovering sets of attributes that discriminate data entities into classes, such as shopping trends for a particular demographic group. In contrast, we are working to develop data mining techniques to discover patterns consisting of complex relationships between entities. Our research is particularly applicable to domains in which the data is event driven, such as counter-terrorism intelligence analysis. In this paper we describe an algorithm designed to operate over relational data received incrementally. Our approach includes a mechanism for summarizing discoveries from previous data increments so that the globally best patterns can be computed by examining only the new data increment. We describe a method by which relational dependencies that span across temporal increment boundaries can be efficiently resolved so that additional pattern instances, which do not reside entirely in a single data increment, can be discovered.

Introduction

Much of current data mining research is focused on algorithms that can discover sets of attributes that discriminate data entities into classes, such as shopping or banking trends for a particular demographic group. In contrast, our work is focused on data mining techniques to discover relationships between entities. Our work is particularly applicable to problems where the data is event driven, such as the types of intelligence analysis performed by counter-terrorism organizations. Such problems require discovery of relational patterns between the events in the environment so that these patterns can be exploited for the purposes of prediction and action.

Also common to these domains is the continuous nature of the discovery problems. For example, Intelligence Analysts often monitor particular regions of the world or focus on long-term problems like Nuclear Proliferation over the course of many years. To assist in such tasks, we are developing data mining techniques that can operate with data that is received incrementally.

In this paper we present Incremental Subdue (ISubdue), which is the result of our efforts to develop an incremental discovery algorithm capable of evaluating data received incrementally. ISubdue iteratively discovers and refines a set of canonical patterns, considered to be most representative of the accumulated data.

Structure Discovery

The work we describe in this paper is based upon Subdue (Holder et al. 2002), which is a graph-based data mining system designed to discover common structures from relational data. Subdue represents data in graph form and

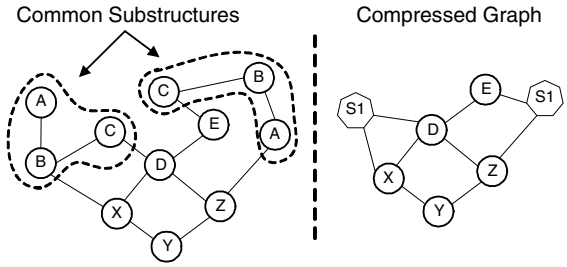

Figure 1. Subdue discovers common substructures within relational data by evaluating their ability to compress the graph.

can support either directed or undirected edges. Subdue operates by evaluating potential substructures for their ability to compress the entire graph, as illustrated in Figure 1. The better a particular substructure describes a graph, the more the graph will be compressed by replacing that substructure with a placeholder. Repeated iterations will discover additional substructures, potentially those that are hierarchical, containing previously compressed substructures.

Subdue uses the Minimum Description Length Principle (Rissanen 1989) as the metric by which graph compression is evaluated. Subdue is also capable of using an inexact graph match parameter to evaluate substructure matches so that slight deviations between two patterns can be considered as the same pattern.

Incremental Discovery

For our work on ISubdue, we assume that data is received in incremental blocks. Repeatedly reprocessing the accumulated graph after receiving each new increment would be intractable because of the combinatoric nature of substructure evaluation, so instead we wish to develop methods to incrementally refine the substructure discoveries with a minimal amount of reexamination of old data.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2005		**00-00-2005 to 00-00-2005**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Structure Discovery in Sequentially Connected Data	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **The University of Texas at Arlington,Department of Computer Science and Engineering,Box 19015,Arlington,TX,76019**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**6**	

Independent Data

In our previous work (Coble et al, 2005), we developed a method for incrementally determining the best substructures within sequential data where each new increment is a distinct graph structure independent of previous increments. The accumulation of these increments is viewed as one large but disconnected graph.

We often encounter a situation where local applications of Subdue to the individual data increments will yield a set of locally-best substructures that are not the globally best substructures that would be found if the data could be evaluated as one aggregate block. To overcome this problem, we introduced a summarization metric, maintained from each incremental application of Subdue, that allows us to derive the globally best substructure without reapplying Subdue to the accumulated data.

To accomplish this goal, we rely on a few artifacts of Subdue's discovery algorithm. First, Subdue creates a list of the n best substructures discovered from any dataset, where *n* is configurable by the user. .

$$Compression = \frac{DL(S) + DL(G \mid S)}{DL(G)} \qquad \text{Eq. 1}$$

Second, we use the value metric Subdue maintains for each substructure. Subdue measures graph compression with the Minimum Description Length principle as illustrated in Equation 1, where *DL(S)* is the description length of the substructure being evaluated, *DL(G|S)* is the description length of the graph as compressed by the substructure, and *DL(G)* is the description length of the original graph. The better our substructure performs, the smaller the compression ratio will be. For the purposes of our research, we have used a simple description length measure for graphs (and substructures) consisting of the number of vertices plus the number of edges. C.f. (Cook and Holder 1994) for a full discussion of Subdue's MDL graph encoding algorithm.

Subdue's evaluation algorithm ranks the best substructure by measuring the inverse of the compression value in Equation 1. Favoring larger values serves to pick a substructure that minimizes *DL(S) + DL(G|S),* which means we have found the most descriptive substructure.

For ISubdue, we must use a modified version of the compression metric to find the globally best substructure, illustrated in Equation 2.

$$Compress_m(S_i) = \frac{DL(S_i) + \sum_{j=1}^{m} DL(G_j \mid S_i)}{\sum_{j=1}^{m} DL(G_j)} \qquad \text{Eq. 2}$$

With Equation 2 we calculate the compression achieved by a particular substructure, S_i, up through and including the current data increment *m*. The $DL(S_i)$ term is the description length of the substructure, S_i, under consideration. The term

$$\sum_{j=1}^{m} DL(G_j \mid S_i)$$

represents the description length of the accumulated graph after it is compressed by the substructure S_i.

Finally, the term

$$\sum_{j=1}^{m} DL(G_j)$$

represents the full description length of the accumulated graph.

$$arg\ max(i) \left[\frac{\sum_{j=1}^{m} DL(G_j)}{DL(S_i) + \sum_{j=1}^{m} DL(G_j \mid S_i)} \right] \qquad \text{Eq. 3}$$

At any point we can then reevaluate the substructures using Equation 3 (inverse of Equation 2), choosing the one with the highest value as globally best.

After running the discovery algorithm over each newly acquired increment, we store the description length metrics for the top n local subs in that increment. By applying our algorithm over all of the stored metrics for each increment, we can then calculate the global top n substructures.

Sequentially Connected Data

We now turn our attention to the challenge of incrementally modifying our knowledge of the most representative pattern when dependencies exist across sequentially received data increments. As each new data increment is received, it may contain new edges that extend from vertices in the new data increment to vertices in previous increments.

Figure 2 illustrates an example where two data increments are introduced over successive time steps.

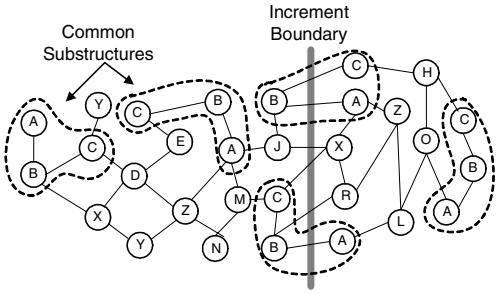

Figure 2. Sequentially connected data

Common substructures have been identified and two instances extend across the increment boundary. Referring back to our counterterrorism example, it is easy to see how analysts would continually receive new information regarding previously identified groups, people, targets, or institutions.

Algorithm

First, we assume certain conditions with respect to the data.

1) The environment producing the data is stable, meaning that the relationships that govern the variables are constant. We will address concept drift in our future work.

2) The pattern instances are distributed consistently throughout the data. We need not rely on a specific statistical distribution. Our requirement is only that any pattern prominent enough to be of interest is consistently supported throughout the data.

Approach

Let

G_n = set of top-n globally best substructures
I_s = set of pattern instances associated with a substructure $s \in G_n$
V_b = set of vertices with an edge spanning the increment boundary and that are potential members of a top-n substructure
S_b = 2-vertex pairs of seed substructure instances with an edge spanning the increment boundary
C_i = set of candidate substructure instances that span the increment boundary and that have the potential of growing into an instance of a top n substructure.

The first step in the discovery process is to apply the algorithm we developed for the independent increments discussed above. This involves running Subdue discovery on the data contained exclusively within the new increment, ignoring the edges that extend to previous increments. We then update the statistics stored with the increment and compute the set of globally best substructures G_n. This process is illustrated in Figure 3.

Based on our defined assumptions, we know that the local data within the new increment is consistent with the rest of the data, so we wish to take advantage of it in forming our knowledge about the set of patterns that are

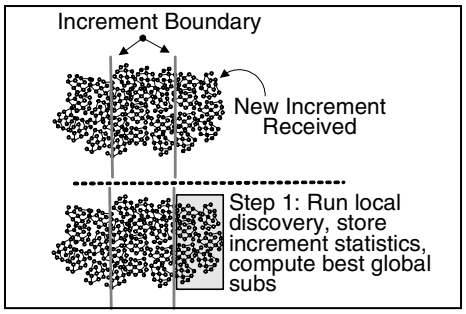

Figure 3. The first step of the sequential discovery process is to evaluate the local data in the new increment

most representative of the system generating the data. Although the set of top-n substructures computed at this point in the algorithm does not consider substructure instances spanning the increment boundary and therefore will not be accurate in terms of the respective strength of the best substructures, it will be more accurate than if we were to ignore the new data entirely prior to addressing the increment boundary.

The second step of our algorithm is to identify the set of boundary vertices, V_b, where each vertex has a *spanning edge* that extends to a previous increment and is potentially a member of one of the top n best substructures in G_n. We can identify all boundary vertices in O(m), where m is the number of edges in the new increment, and then identify those that are potential members of a top-n substructure in O(k), where k is the number of vertices in the set of substructures G_n. Figure 4 illustrates this process.

For the third step we create a set of 2-vertex substructure seed instances by connecting each vertex in V_b with the spanning edge to its corresponding vertex in a previous increment. We immediately discard any instance where the second vertex is not a member of a top-n substructure (all elements of V_b are already members of a top-n

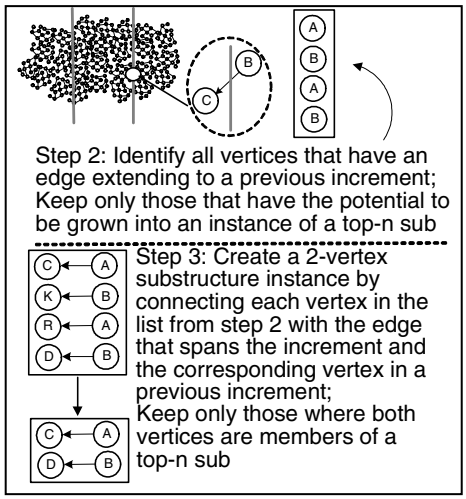

Figure 4. The second step is to identify all boundary vertices that could possibly be part of an instance of a top n pattern. The third step is to create 2-vertex substructure instances by joining the vertices that span the increment boundary.

substructure), which again can be done in O(k). A copy of each seed instance is associated with each top-n substructure, $s_i \in G_n$, for which it is a subset.

To facilitate an efficient process for growing the seed

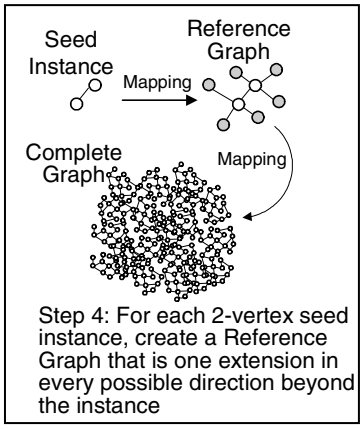

Figure 5. To facilitate efficient instance extension, we create a reference graph, which we keep extended one step ahead of the instances it represents.

instances into potential instances of a top-n substructure, we now create a set of reference graphs. We create one

reference graph for each copy of a seed instance, which is in turn associated with one top-n substructure. Figure 5 illustrates this process. We create the initial reference

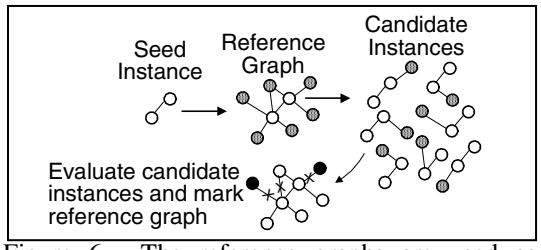

Figure 6. The reference graphs are used as a template to extend new candidate instances for evaluation against the top-n substructures. Failed extensions are propagated back into the reference graph with marked edges and vertices, to guide future extensions.

graph by extending the seed instance by one edge and vertex in all possible directions. We can then extend the seed instance with respect to the reference graph to create a

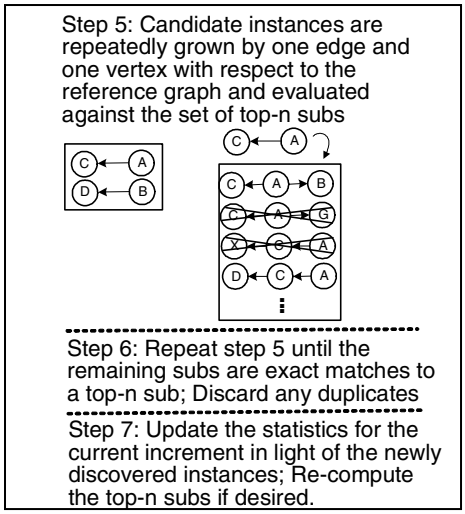

Figure 7. The fifth and sixth steps repeatedly extend the set of seed instances until they are either grown into a substructure from S_t or discarded.

set of candidate instances C_i, for each top-n substructure $s_i \in G_n$, illustrated in Figure 6. The candidate instances represent an extension by a single edge and a single vertex, with one candidate instance being generated for each possible extension beyond the seed instance. We then evaluate each candidate instance, $c_{ij} \in C_i$ and keep only those where c_{ij} is still a subgraph of s_i. For each candidate instance that is found to not be a subgraph of a top-n substructure, we mark the reference graph to indicate the failed edge and possibly a vertex that is a dead end. This prevents redundant exploration in future extensions and significantly prunes the search space.

In the fifth step (Figure 7), we repeatedly extend each instance, $c_{ij} \in C_i$, in all possible directions by one edge and one vertex. When we reach a point where candidate instances remain but all edges and vertices in the reference graph have already been explored, then we again extend the reference graph frontier by one edge and one vertex. After each instance extension we discard any instance in C_i that is no longer a subgraph of a substructure in G_n. Any instance in C_i that is an exact match to a substructure in G_n is added to the instance list for that substructure, I_s, and

```
While Ci ≠ ∅
  For all c ∈ Ci
    found = false
    For all s ∈ Gn
      If c ⊆ s
        found = true
        If c = s
          Is' ⋃ c
    If not found or c = s
      Ci ← Ci - c
  If Ci ≠ ∅
    Ci ← ExtendInstance(Ci)
```

Figure 8. Pseudocode for steps 5 and 6.

removed from C_i. The pseudocode for this process is illustrated in Figure 8.

Once we have exhausted the set of instances in C_i so that they have either been added to a substructure's instance list or discarded, we update the increment statistics to reflect the new instances and then we can recalculate the top-n set, G_n, for the sake of accuracy, or wait until the next increment.

Evaluation

To validate our work, we have conducted two sets of experiments, one on synthetic data and another on data simulated for the counterterrorism domain.

Synthetic Data. Our synthetic data consists of a randomly generated graph segment with vertex labels drawn uniformly from the 26 letters of the alphabet. Vertices have between one and three outgoing edges where the target vertex is selected at random and may reside in a previous data increment, causing the edge to span the increment boundary. In addition to the random segments,

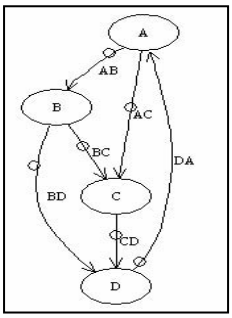

Figure 9. Predefined substructure embedded in synthetic data.

we intersperse multiple instances of a predefined substructure. For the experiments described here, the

predefined substructure we used is depicted in Figure 9. We embed this substructure internal to the increments and also insert instances that span the increment boundary to test that these instances are detected by our discovery algorithm.

Figure 10 illustrates the results for a progression of five experiments. The x-axis indicates the number of increments that were processed and the respective size in terms of vertices and edges. To illustrate the experiment methodology, consider the 15-increment experiment. We provide ISubdue with the 15 increments in sequential order

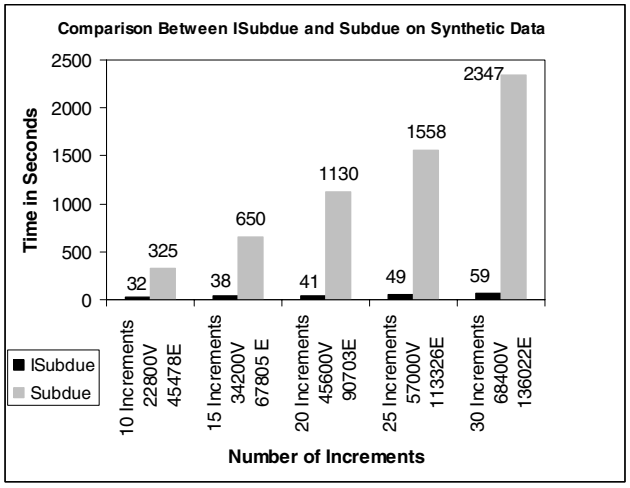

Figure 10. Comparison of ISubdue and Subue on on increasing number of increments for synthetic data.

as fast as the algorithm can process them. The time depicted is for processing all 15 increments. We then aggregate all 15 increments and process them with Subdue for the comparison. The five results shown in Figure 10 are not cumulative, meaning that each experiment includes a new set of increments. It is reasonable to suggest then that adding five new increments – from 15 to 20 – would require approximately three additional seconds of processing time for ISubdue, whereas Subdue would require the full 1130 seconds because of the need to reprocess all of the accumulated data.

Counterterrorism Data. The counterterrorism data was generated by a simulator created as part of the Evidence Assessment, Grouping, Linking, and Evaluation (EAGLE) program, sponsored by the U.S. Air Force Research Laboratory. The simulator was created by a program participant after extensive interviews with Intelligence Analysts and several studies with respect to appropriate ratios of noise and clutter. The data we use for discovery represents the activities of terrorist organizations as they attempt to exploit vulnerable targets, represented by the execution of five different event types. They are:

Two-way-Communication: Involves one initiating person and one responding person.
N-way-Communication: Involves one initiating person and multiple respondents.
Generalized-Transfer: One person transfers a resource.
Applying-Capability: One person applies a capability to a target
Applying-Resource: One person applies a resource to a target

The data also involves targets and groups, groups being comprised of member agents who are the participants in

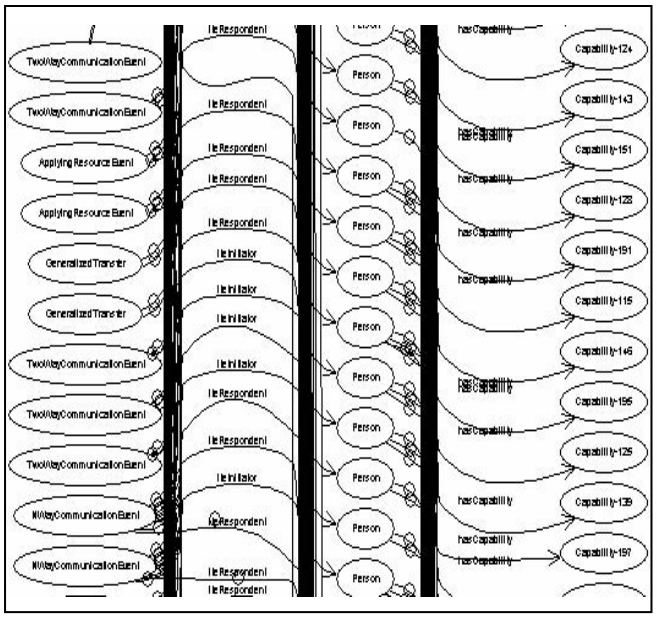

Figure 11. A section of the graph representation of the counterterrorism data used for our evaluation.

the aforementioned events. All data is generalized so that no specific names are used. Figure 11 illustrates a small cross-section of the data used in our experiments.

The intent of this experiment was to evaluate the performance of our research on ISubdue against the

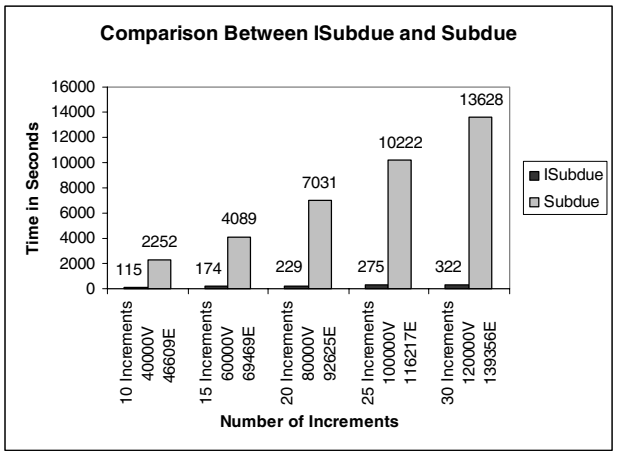

Figure 12. Comparison of run-times for ISubdue and Subdue on increasing numbers of increments for counterterrorism data.

performance of the original Subdue algorithm. We are interested in measuring performance along two dimensions, run-time and the best reported substructures.

Figure 12 illustrates the comparative run-time performance of ISubdue and Subdue on the same data. As for the synthetic data, ISubdue processes all increments

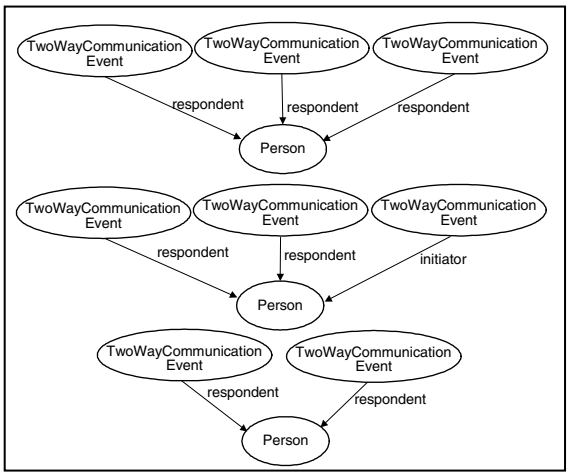

Figure 13. The top 3 substructures discovered by both ISubdue and Subdue for the counterterrorism data.

successively whereas Subdue batch processes an aggregation of the increments for the comparative result.

Figure 13 depicts the top three substructures discovered by both ISubdue and Subdue. This set of substructures was consistently discovered for all five experiments introduced in Figure 12.

Conclusions and Future Work

In this paper we have presented a method for mining graph-based data received incrementally over time. We have demonstrated that our approach provides a significant savings, in terms of processing time, without sacrificing accuracy. This work provides essential capabilities necessary for the next phase of our research in which we will investigate the notion of drifting concepts, which is a significant challenge for time-sequenced data.

Concept Drift

In the traditional machine learning problem (Mitchell, 2004; Vapnik, 1995), it is generally stated that some stable function F(x) is generating an attribute vector x. The attribute vector x represents the observable features of the problem space. This definition extends intuitively to data mining. However, in sequential discovery problems, the domains are such that the underlying relationships between system variables often change over time. Referring back to our counter-terrorism domain, it is certainly the case that terrorist organizations change their behaviors in unpredictable ways and adapt to counter-terrorism efforts. There are approaches to machine learning in the presence of shifting concepts, such as the sliding window approach (Widmer & Kubat, 1996), where only the last *n* data points are used to update the learned model, but such approaches are often naïve in the sense that they disregard valuable information learned outside of the data window. This is akin to forgetting everything discovered about a terrorist organization's behaviors and capabilities when in fact only a small portion of their behaviors have changed, like an alteration in communication patterns. Our future work will focus on developing methods for structure discovery when the underlying system is undergoing change.

Increment Size

We have learned from our experimentation that the size of the data increments must be chosen carefully. If data increments are too small, then the local discovery process we use as a precursor to our boundary evaluation may be overly biased to incomplete substructures. In practice, it is often possible to select an appropriately sized increment boundary given some knowledge about the domain. However, there are situations where the data may obey irregular cycles and therefore the increment size shouldn't be set to a fixed size. In our future work we intend to explore statistical and information theoretic measures for dynamically selecting an increment size.

Acknowledgements

This research is sponsored by the Air Force Research Laboratory (AFRL) under contract F30602-01-2-0570. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of AFRL or the United States Government.

References

1. Coble, J., Rathi, R., Cook, D., Holder, L. *Iterative Structure Discovery in Graph-Based Data*. To appear in the International Journal of Artificial Intelligence Tools, 2005.
2. Cook, D. and Holder, L. 1994. *Substructure Discovery Using Minimum Description Length and Background Knowledge*. In Journal of Artificial Intelligence Research, Volume 1, pages 231-255.
3. Holder, L., Cook, D., Gonzalez, J., and Jonyer, I. 2002. *Structural Pattern Recognition in Graphs*. In Pattern Recognition and String Matching, Chen, D. and Cheng, X. eds. Kluwer Academic Publishers.
4. Mitchell, T. *Machine Learning*, McGraw Hill, 1997.
5. Rissanen, J. 1989. *Stochastic Complexity in Statistical Inquiry*. World Scientific Publishing Company, 1989.
6. Vapnik, V. *The Nature of Statistical Learning Theory*, Springer, New York, NY, USA, 1995.
7. Widmer, G.. and Kubat, M. . *Learning in the Presence of Concept Drift and Hidden Contexts*. Machine Learning, 23, 69-101, 1996.